

# Визуализация отзывов, облако слов на Python

Касьянов Артём Владимирович  
НБИбд-01-21

# Введение

В современном мире объемы текстовых данных стремительно растут, и для эффективного анализа таких данных требуется использовать мощные инструменты и методы. Одним из популярных подходов к визуализации текстовой информации является создание облака слов. Облако слов позволяет наглядно представить наиболее частые слова в тексте, что помогает быстро выявить основные темы и тенденции. Цель данной презентации — продемонстрировать, как с помощью языка программирования Python и его библиотек можно создать интерактивное облако слов из текстовых данных. Мы рассмотрим процесс от предварительной обработки текста до визуализации результатов.



# С о д е р ж а н и е

## 1 – Цель создания облака

Наглядное представление частотности слов

## 3 – Разбор кода, взаимодействия

Обработка текста, подсчёт частот, визуализация

## 2 – Используемые библиотеки

rumorphy2, NLTK, pandas, WordCloud, Matplotlib, Tkinter

## 4 – Демонстрация и вопросы

Показ работы программы, ответы на вопросы

# 1 Цель создания облака

## Преимущества:

- Интуитивно понятное восприятие: Легко определить наиболее важные слова и темы.
- Эффективный анализ: Позволяет быстро анализировать большие объемы текста.
- Визуальная привлекательность: Привлекает внимание и делает данные более доступными для восприятия.
- Интерактивность: В нашем проекте облако слов можно настраивать, изменяя количество отображаемых слов.

# Тесла





# 2 Используемые библиотеки



Библиотеки Python — это файлы с шаблонами кода. Их создали для того, чтобы люди не набирали каждый раз заново один и тот же код: достаточно открыть файл, вставить свои данные и получить результат.

Библиотеки:

- Rymorphy2 — это морфологический анализатор для русского языка.
- NLTK (Natural Language Toolkit) — это библиотека для работы с текстом и естественным языком.
- Pandas — это библиотека для работы с табличными данными.
- WordCloud — это библиотека для генерации облака слов.
- Matplotlib — это библиотека для визуализации данных.
- Tkinter — это стандартная библиотека Python для создания графического интерфейса пользователя (GUI).

# 3 Разбор кода подгрузка

Подгрузка

```
# Загрузка необходимых ресурсов NLTK
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger_ru')

# Загрузка данных, пропуская первые 13 строк
data = pd.read_excel(io='D:\SL датасеты РУДН\Мармелад.xlsx', skiprows=13)

# Удаление столбцов, в которых все значения пустые
data.dropna(axis=1, how='all', inplace=True)

# Инициализация морфологического анализатора
morph = pymorphy2.MorphAnalyzer()
```

Электромагнитная индукция

```
# Загрузка стоп-слов и добавление дополнительных слов для удаления
stop_words = set(stopwords.words('russian'))
additional_stop_words = {
    "текст", "изображение", "haribo", "mamba", "mayama", "bonpari", "bebeto", "chips chups", "krut frut", "kinder",
    "nestle", "lays", "нюша", "южный", "sweetlife", "весь", "ип", "продавец", "производитель", "товар",
    "отзыв", "оценка", "состав", "ещё", "весь", "чупа", "покупка", "шайхлислам", "чупс",
    "наш", "энэкт", "день", "магазин", "покупатель", "комментарий", "недостаток", "покупатель", "вкус", "мармелад",
    "покупка", "мнение", "бренд", "достоинство", "мармеладка", "упаковка", "общество", "ответственность", "юнэкт",
    "юниона", "ooo", "прийти", "мочь", "артикул", "казахстан", "мелла", "свой", "перфетти", "владимирович", "который",
    "пачка",
    "индивидуальный", "вид", "предприниматель", "оляга", "видеть", "ваш", "это", "самый", "срок", "россия", "год",
    "слапогузов"
}
stop_words.update(additional_stop_words)
```

Мрф. анализатор

```
word = 'бегающий'
parsed_word = morph.parse(word)[0]
print(parsed_word.normal_form) # Выведет: бегать
print(parsed_word.tag.POS)    # Выведет: VERB (глагол)
```

# 3 Разбор кода функции

## 1ая фильтрация

```
# Функция для очистки текста
1 usage
def clean_text(text):
    text = str(text) # Убедимся, что входной текст это строка
    text = re.sub(pattern: r'http\S+', repl: '', text) # Удаление URL-адресов
    text = re.sub(pattern: r'\d+', repl: '', text) # Удаление цифр
    text = re.sub(pattern: r'[^\А-яЁё]+', repl: ' ', text) # Удаление всего, что не входит в кириллический алфавит
    text = text.lower() # Приведение текста к нижнему регистру
    return text
```

## 2ая фильтрация

```
# Функция для лемматизации текста и удаления ненужных частей речи
1 usage
def lemmatize_and_filter(text):
    words = word_tokenize(text)
    filtered_words = []
    for word in words:
        lemma = morph.parse(word)[0].normal_form
        pos_tag = morph.parse(word)[0].tag.POS
        if (lemma not in stop_words and pos_tag and
            pos_tag not in {'INTJ', 'PRCL', 'CONJ', 'PREP'} and len(lemma) > 1):
            filtered_words.append(lemma)
    return ' '.join(filtered_words)
```

## Применение (сообщение)

```
# Полная предобработка текста
1 usage
def preprocess_text(text):
    text = clean_text(text)
    text = lemmatize_and_filter(text)
    return text
```

```
# Применение предобработки к данным
data['Очищенный текст'] = data['Сообщение'].apply(preprocess_text)
```

# 3 Разбор кода функции

## 1ая фильтрация

```
# Применение предобработки к данным
data['Очищенный текст'] = data['Сообщение'].apply(preprocess_text)

# Фильтрация данных по тональности
positive_data = data[data['Тональность'] == 'позитивная']['Очищенный текст']
negative_data = data[data['Тональность'] == 'негативная']['Очищенный текст']
neutral_data = data[(data['Тональность'] == 'нейтральная']]['Очищенный текст']

# Подсчёт частоты слов для позитивных и негативных отзывов
positive_word_counts = Counter(" ".join(positive_data).split())
negative_word_counts = Counter(" ".join(negative_data).split())
neutral_word_counts = Counter(" ".join(neutral_data).split())

# Объединение словарей с частотой слов
word_counts = positive_word_counts + negative_word_counts + neutral_word_counts
```

## функция цвета

```
# Создание функции для цветового кодирования
1 usage
def get_color(word):
    pos_count = positive_word_counts[word]
    neg_count = negative_word_counts[word]
    neu_count = neutral_word_counts[word]
    total_count = pos_count + neg_count + neu_count
    pos_ratio = pos_count / total_count
    neg_ratio = neg_count / total_count
    neu_ratio = neu_count / total_count

    # Определение цвета на основе частоты появления слова в каждой группе
    r = int(255 * neg_ratio)
    g = int(255 * pos_ratio)
    b = int(255 * neu_ratio)
    return f"rgb({r},{g},{b})"
```

## применение, вывод

```
# Функция для генерации облака слов
1 usage
def generate_wordcloud(num_words):
    top_words = word_counts.most_common(num_words)
    wordcloud = WordCloud(width=800, height=400, color_func=lambda *args,
        **kwargs: get_color(args[0])).generate_from_frequencies(dict(top_words))

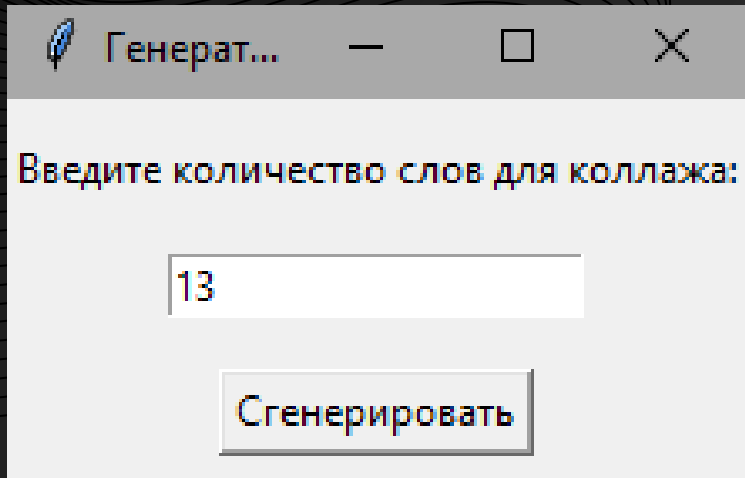
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.show()
```



# 3 Разбор кода функции

Tkinter

Ввод



Генерат...

Введите количество слов для коллажа:

13

Сгенерировать

ИТОГ

```
# Создание GUI с помощью Tkinter
1 usage
def create_gui():
    def on_generate():
        try:
            num_words = int(entry.get())
            generate_wordcloud(num_words)
        except ValueError:
            messagebox.showerror( title="Ошибка", message="Пожалуйста, введите корректное число.")

    root = tk.Tk()
    root.title("Генератор Облака Слов")

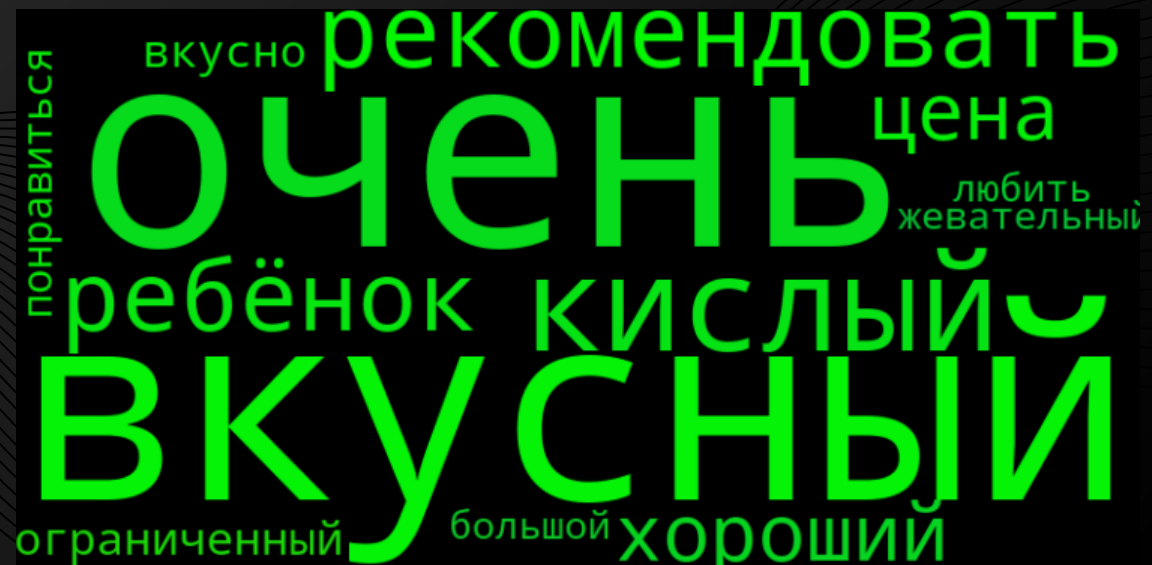
    label = tk.Label(root, text="Введите количество слов для коллажа:")
    label.pack(pady=10)

    entry = tk.Entry(root)
    entry.pack(pady=5)

    button = tk.Button(root, text="Сгенерировать", command=on_generate)
    button.pack(pady=10)

    root.mainloop()

# Запуск GUI
create_gui()
```



ОЧЕНЬ

ВКУСНЫЙ

рекомендовать

кислый

ребёнок

хороший

популярный

ограниченный

большой

любить

жевательный

вкусно

цена

# 4 Демонстрация и итоги



## Итоги

Этот проект демонстрирует мощь и гибкость Python для анализа и визуализации текстовых данных. Мы рассмотрели, как загружать и предобрабатывать данные, очищать и лемматизировать текст, а также как визуализировать результаты в виде облака слов. Интерактивный интерфейс с использованием Tkinter позволяет пользователю легко настраивать параметры визуализации, что делает проект не только полезным, но и удобным в использовании.

**Спасибо за внимание**

The bottom of the slide features a decorative graphic consisting of multiple thin, white, wavy lines that flow horizontally across the frame. These lines are set against a background that transitions from a light purple on the left to a light blue on the right. The overall aesthetic is clean and modern.